REMARKS

Claims 1, 4-12, 31-39, 42-44, 46-51, 54-56, and 58-62 remain pending in the application. Reconsideration is respectfully requested in light of the following remarks.

Section 103(a) Rejection:

The Examiner rejected claims 1, 4-12, 31-39, 42-44, 46-51, 54-56 and 58-62 under 35 U.S.C. § 103(a) as being unpatentable over "The Error Handling Interface (H5E)" (hereinafter "H5E-A") in view of Srivastava et al. (U.S. Publication 2005/0160431) (hereinafter "Srivastava"). Applicants respectfully traverse this rejection for at least the following reasons.

In regard to claim 1, the cited art does not teach or suggest *in each of two or more threads of a multithreaded program, for each error generated by one or more functions executed in the thread, store an error trace element in a memory storage area private to the thread in accordance with an application programming interface (API) to an error trace mechanism.* As illustrated in Fig. 2 and described in the accompanying description (e.g., paragraph [0021]) of Applicant's specification, each of the two or more threads in the multithreaded program is associated with a particular, separate and distinct, "thread private data", or "private data area", for the thread, which is exemplary of a memory storage area private to the corresponding thread. Error trace elements generated for a particular thread are stored to the corresponding private data area, a memory storage area private to the thread, as is clearly illustrated in Fig. 2 and disclosed in paragraph [0021]:

> Each thread private data 100 is a storage area (memory) specific to a particular thread. Each thread may store and access data in its associated thread private data 100 for the thread. In one embodiment, when error traces are stored using thread private data, all error trace elements generated for a particular thread are recorded in the thread's private data area. Thus, each thread private data 100 may store error traces 104, if any, for its associated thread.

The H5E-A reference does not teach or suggest an error trace mechanism for a multithreaded program configured to, for each error generated by one or more functions executed in each of **two or more threads** in the multithreaded program, store an error trace element in a memory storage area **private to** the corresponding thread, as is recited in claim 1. Further, the H5E-A reference does not teach or suggest an error trace mechanism configured to obtain an error trace for each of **two or more threads** of a multithreaded program, wherein each error trace includes one or more error trace elements specific to the corresponding thread, as is recited in claim 1.

The Examiner asserts Srivastava discloses "two or more threads of a multithreaded program and storing an error trace element in a memory area private to the thread for each of the two or more threads," citing paragraph [0003], "Each distributed node may also maintain multiple trace logs corresponding to separate threads on that node." However, contrary to the Examiner's assertion, Srivastava, alone or in combination with the other reference, does not teach the subject matter as recited in the claim, for at least the following reasons.

Paragraph [0003] states (emphasis added): "Often, each of the distributed nodes maintains a separate log file to store traces for their respective threads. Each distributed node may also maintain multiple trace logs corresponding to separate threads on that node." It is clear from this paragraph that what Srivastava is describing are trace log files maintained by and on the distributed nodes. This is further made clear by the next paragraph, which begins "Diagnosing problems using multiple trace logs often involves a **manual** process of repeatedly inspecting different sets of the trace logs." Thus, Srivastava's "trace logs" are clearly log files maintained on and by the distributed nodes. Srivastava's "trace logs" are clearly not the same "memory storage areas private to the threads", as recited in Applicants' claim 1, but are instead trace log files maintained by the distributed nodes, independently of the threads themselves. The log files in Srivastava are not private to any particular thread running on one of one of Srivastava's nodes. In fact, Srivastava teaches that a given log file stores traces for all threads on the corresponding node.

In addition, Srivastava's "trace logs" record messages output by statements inserted in the application code by a programmer, as indicated in paragraph [0001]:

> Tracing is an approach for logging the state of computer applications at different points during its course of execution. Tracing is normally implemented by inserting statements in the computer application code that outputs status/state messages ("traces") as the statements are encountered during the execution of the code. Statements to generate traces are purposely placed in the computer application code to generate traces corresponding to activities of interest performed by specific sections of the code. The generated trace messages can be collected and stored during the execution of the application to form a trace log.

In contrast to Srivastava's teaching of log files maintained on and by distributed nodes that record messages output by statements inserted in the application code by a programmer, claim 1 recites *in each of two or more threads of a multithreaded program, for each error generated by one or more functions executed in the thread, store an error trace element in a memory storage area private to the thread in accordance with an application programming interface (API) to an error trace mechanism*. H5E-A and Srivastava clearly do not teach the limitations as recited in claim 1.

In addition, Srivastava does not teach that the "trace logs" are error traces as recited in Applicants' claim 1. Instead, the trace logs in Srivastava log information about the normal operation of a node, not errors. As noted above, Srivastava only teaches trace log files that are formed by collecting and storing various trace messages ("status/state messages ("traces") as the statements are encountered during the execution of the code"). Srivastava teaches, in paragraph [0002], that "Programmers often use tracing and trace logs to diagnose problems or errors that arise during the execution of a computer application. When such a problem or error is encountered, trace logs are analyzed to correlate trace messages with the application code to determine the sequence, origin, and effects of different events in the systems and how they impact each other. This process allows analysis/diagnoses of unexpected behavior or programming errors that cause problems in the application code." Applicants note that this paragraph, in the background section, contains the only mentions of "error" in the entire Srivastava reference. Note

that Srivastava does not state that the errors themselves are logged. From this teaching in paragraph [0002], it is clear that, in contrast to teaching an error trace including error trace elements each including information describing a particular error generated during execution of the corresponding thread, Srivastava teaches a trace log file including various messages that must be manually "analyzed" to diagnose "unexpected behavior or programming errors that cause problems in the application code."

The Examiner relies on Srivastava, paragraph [0003], which states "Each distributed node may also maintain multiple trace logs corresponding to separate threads on that node," to teach "storing an error trace element in a memory storage area private to the thread for each of two or more threads." **However, simply because a trace log corresponds to a particular thread does not mean that it is stored in a private memory area for that thread.**

Moreover, paragraph [0003] is in Srivastava's background section, and describes a method of a node maintaining multiple trace log files corresponding to separate threads. This method is prior art to Srivastava's method and is not how Srivastava's system works. **In fact, Srivastava actually teaches away from the notion of multiple trace logs corresponding to separate threads and private to the threads.** See, for example, paragraph [0004], which describes problems with the conventional approach described in paragraph [0003], including "Each distributed node may also maintain multiple trace logs corresponding to separate threads on that node," as relied upon by the Examiner. The very purpose of Srivastava is to provide methods so that the "multiple trace logs" are linked and thus not private to the threads. **Using Srivastava's method, the "trace logs" clearly are not and cannot be said to be "private to the threads."** This is made clear, for example, in the Abstract, repeated in paragraph 7, which states (emphasis added): "when tracing a series of related events that span across a plurality of threads, a token may be passed from one thread to another, thereby allowing a link between the threads to be marked within the one or more traces."

Moreover, claim 1 of the instant application further recites *obtaining an error*

trace for each of the two or more threads of the multithreaded program in accordance with the API to the error trace mechanism, wherein each error trace includes one or more error trace elements specific to the corresponding thread. The very purpose of Srivastava's method is to provide links between the threads to be marked within the one or more trace logs, and thus one of Srivastava's trace logs would purposefully include information that is not specific to the corresponding thread. Thus, Srivastava's disclosed method is counter to, and in contrast with, the subject matter as recited in Applicants' claim1.

Furthermore, the Examiner has not stated a proper reason to combine the teachings of the cited art. The Examiner asserts "it would have been obvious…to incorporate the teachings of Srivastava into the teachings of H5E-A" because one of ordinary skill in the art would "want to be able to debug related events that span across a plurality of threads as suggested by Srivastava." First, the reason provided by the Examiner is not commensurate with the feature from the Srivastava reference that the Examiner proposes to combine with H5E-A to result in the claimed invention. The reason from the Abstract, quoted by the Examiner, relates to Srivastava's proposed method of "a token may be passed from one thread to another, thereby allowing a link between the threads to be marked within the one or more traces." However, the Examiner has not actually proposed that this feature be combined with H5E-A. Moreover, if Srivastava were combined with H5E-A for the reason given by the Examiner, then the resultant trace logs would clearly not be "private to the threads" and would not be composed of trace elements specific to the corresponding thread. In addition, as noted above, Srivastava actually teaches away from the notion of multiple trace logs corresponding to separate threads and private to the threads. Combining Srivastava with H5E-A, if possible, would if anything only result in H5E-A incorporating Srivastava's trace log files maintained by nodes and including links to other trace logs, which thus could not be said to be "private to the threads" as recited in Applicants' claim 1.

Furthermore, there is no enabling description in the cited art for a multithreaded embodiment of the teachings of the H5E-A that would be achieved by such a combination, if the combination were possible. The references do not describe how a multi-threaded version of the teachings of H5E-A would work, or in any way provided an enabling disclosure for a multi-threaded version of the teachings of H5E-A. "In order to render a claimed apparatus or method obvious, the prior art must enable one skilled in the art to make and use the apparatus or method." *Motorola, Inc. v. Interdigital Tech. Corp.*, 43 USPQ2d 1481, 1489 (Fed. Cir. 1997) (quoting *Beckman Instruments, Inc. v. LKB Produkter AB*, 13 USPQ2d 1301, 1304 (Fed. Cir. 1989)); see also *Rockwell Int'l Corp. v. United States*, 47 USPQ2d 1027, 1032 (Fed. Cir. 1998). **Since the cited art does not enable a multi-threaded version of its teachings, the rejection is improper.**

Thus, one of ordinary skill would not have combined the teachings of Srivastava with the teachings of H5E-A in the manner proposed by the Examiner. Accordingly, the Examiner has failed to establish a *prima facie* case of obviousness.

Thus, for at least the reasons presented above, the rejection of claim 1 is not supported by the cited prior art and removal thereof is respectfully requested. Similar remarks as those above regarding claim 1 also apply to claims 9, 31, 36, 39, 47, 51, and 59.

Applicants also assert that the rejection of numerous ones of the dependent claims is further unsupported by the cited art. However, since the rejection has been shown to be unsupported for the independent claims, a further discussion of the dependent claims is not necessary at this time.

CONCLUSION

Applicants submit the application is in condition for allowance, and notice to that effect is respectfully requested.

If any fees are due, the Commissioner is authorized to charge said fees to Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C. Deposit Account No. 501505/5681-69401/RCK.

Respectfully submitted,

/Robert C. Kowert/
Robert C. Kowert, Reg. #39,255
Attorney for Applicants

Meyertons, Hood, Kivlin, Kowert, & Goetzel, P.C.
P.O. Box 398
Austin, TX 78767-0398
Phone: (512) 853-8850

Date: February 12, 2009